UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/544,512	04/06/2000	Corneliu I. Lupu	MSFT114614	9057

26389 7590 12/01/2004

CHRISTENSEN, O'CONNOR, JOHNSON, KINDNESS, PLLC
1420 FIFTH AVENUE
SUITE 2800
SEATTLE, WA 98101-2347

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 12/01/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/544,512	LUPU ET AL.	
	Examiner	Art Unit	
	Tuan A Vu	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 August 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3-7,9-13 and 15-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3-7,9-13 and 15-18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's communication filed 8/10/2004.

As indicated in Applicant's communication, claims 1, 5, 7, 11-13, and 17-18 have been amended and claims 2, 8, and 14 canceled. Claims 1, 3-7, 9-13, 15-18 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1, 3-7, 9-13, 15-18 are rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 1-36 of U.S. Patent No. 6,745,385 (hereinafter '385) in view of Edwards et al., USPN: 5,901,315 (hereinafter Edwards).

As per claim 1, '385 claim 1 also recites a method for patching an application including determining whether the application is compatible with a computer operating system; and if it is determined that the application is incompatible, determining the points that need to be patched, running the application and patching the application at those points. But '385 does not recite when determining that the application is incompatible, starting a debugger to run the application and setting at least one breakpoint within said application indicating a stopping point. However,

'385 recites inserting a dynamic linked library at the application address space, such DLL determining which functions needing patch and performing the patching. Analogous to '385's use of DLL for fixing a bug with a patch, Edwards, in a method using a runtime API to debug an application, discloses a debugger that is started and includes native methods and DLLs and setup of breakpoint therewith (Fig. 2, col. 3, lines 54-58; col. 4, lines 2-14; *breakpoint* – Fig. 4). It would have been obvious that one skill in the art would be motivated to modify the DLL-based patching of runtime application by '385 by providing a debug engine as taught by Edwards because a runtime debug package as by Edwards in which native routines as well as DLLs provided as a whole can alleviate piecemeal calls which would be resources consuming (see Edwards BACKGROUND). In light the debugger as taught by Edwards, the limitation as to set a breakpoint as stopping point would have been also obvious because the concept of setting breakpoints while running a debugger is a known concept as well as the establishing of point at which the debugger should stop, because this is inherent to any runtime call invoking an external program for testing, instrumenting or patching purpose.

As per claims 7 and 13, for corresponding to instant claim 1, are also unpatentable over '385 claims 1, 8, 14, 20, 26 and 32, for all those claims recite the same limitations as in '385 claim 1.

As per claims 3-4, and 15-16, these are unpatentable over '385 claims 3-4, 10-11, 22-23, 27-28, 34-35 for these recite the virtually identical limitations as those therein.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 5-7, 11-13, and 17-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stone, "In memory patching: three approaches", *(how to introduce breakpoints in an automated debugger and other marvels)*, March 1998, <http://fravia.anticrack.de/stone1.htm>, (hereinafter Stone), in view of Lillich, USPN: 5,619,698 (hereinafter Lillich)

As per claim 1, Straub discloses a method for patching a computer application program comprising:

starting a debugger (e.g. *loader, CreateProcessA, API-hook* – pg. 4, bottom pg. 7, top; pg. 8, bottom pg. 9, top) to run the incompatible application indicating a stopping point for the debugger (e.g. pg. 8, 2nd , 3rd para - Note: running a API hook with Event detection inherently includes a start and stopping breakpoint);

running the steps of the incompatible application through the debugger (pg. 8-10); and patching the application (pg. 8-10)

But Stone does not explicitly disclose determining whether or not the computer application program is compatible with a computer operating system executing the application (e.g. col. 4, line 4 to col. 5, line 27); and if the computer application program is determined to be incompatible starting up a debugger. But the purpose of patching in itself signifies avoiding executing a portion of an application thus entails obviating executing part of the program that is considered not compatible with the executing environment; hence the notion of incompatibility

to the runtime operating system is strongly implied. However, Stone shows identifying when patches is appropriate with respect to OS compatibility and mentions about invoking programs to provide patches for the addressing space of the target system whenever operating system switches occurs (e.g. ...*when patching is appropriate* - pg. 2, 2nd para to pg. 3, top 2 para), i.e. code instrumentation via a debugger to accommodate potential OS compatibility issue. Modifying application memory in order to accommodate for OS discrepancies or to apply patch related thereto was a known concept for patch upgrade at the time the invention was made. In a method to correct application code executing potentially under a different operating system analogous to the code patching by Straub, Lillich discloses selective adding of patches by an operating system (e.g. col. 6, lines 7-48), dynamic link libraries leading to patches for addressing potentially incompatible application (e.g. Fig. 4), calling upon a combination of analysis of fragment (e.g. DLLs) of application to determine if patches are required (e.g. Fig. 7, 10). It would have been obvious for one of ordinary skill in the art at the time the invention was made to apply a use of patching debugger as suggested by Stone in matching OS switching instances and apply determining methods for checking application for need of patches as taught by Lillich, because this would provide flexibility and correctness to the application of patches for an given operating system, thus obviate fallouts/backfires due to virtually rigid and undifferentiated installation of patches without prior justification (Lillich: col. 5, lines 21-54).

Nor does Stone explicitly disclose patching the incompatible application whenever a breakpoint has been reached. In view of the breakpoints taught from above (pg. 8, 2nd , 3rd para) along with the loading of the DLL to effect a patch, it would have been obvious for one skill in the art at the time the invention was made to provide breakpoints while running the debugger as

taught by Stone so as to be able to patch the application based thereupon, the concept of providing breakpoints at critical part of code to patch being a known concept, and by applying this practice to the debugger by Stone would enable patching at the determined areas of code and thus alleviate excessive use of runtime resources.

As per claim 5, Stone (in combination with Lillich) discloses loading and executing a debugger containing a set breakpoints, each breakpoint having a set of instructions for patching the application (e.g. pg. 4, bottom pg. 7, top; pg. 8, bottom pg. 9, top); the debugger setting a set breakpoints within the application (see claim 1). Stone does not explicitly teach a debugger containing a set of breakpoints and accessing it for setting the breakpoints within the incompatible application. But the fact that a debugger is set to run necessarily entails user's presetting of breakpoints which are purported to be accessed; and the notion that breakpoints are structurally stored in the debugger's portion of the runtime memory is equated to a code representation of list of breakpoints; hence the limitation of storing a list not explicitly disclosed by Stone is implicitly disclosed.

As per claim 6, Stone discloses handle for any process dynamically invoked during the patching (see pg. 4, 2nd para) but does not expressly disclose calling a handler to handle a patch at a given breakpoint, even though a debugger is known for detecting a breakpoint events as suggested from Stone. But official notice is taken that using a special purpose handler at particular point, or breakpoints (if example is needed: see Edwards, Fig. 4) identified by a debugger or any patching API was a known concept at the time the invention was made. Hence, based on such common practice, the calling of a handler at the breakpoint event for performing a patch would have been implicitly disclosed or obvious because one skill in the art would have been

motivated to provide a patch handler to Stone' s dynamic process because first, a debugger is made to address breakpoints or watchpoints, and second, by using a handler memory resources would not be not retained once the patching call exits.

As per claim 7, this is the computer-readable medium version of claim 1 above, hence incorporates the rejection thereof for the same obvious reasons.

As per claim 11, this is the computer-readable medium version of claim 5, hence incorporates the corresponding rejections set forth in therein for the same reasons.

As per claim 12, this is the computer-readable medium version of claim 6 above; hence incorporates the corresponding rejections set forth in therein for the same reasons.

As per claim 13, this is the system version of claim 1 above, hence incorporates the rejection thereof for the same obvious reasons.

As per claim 17, this is the system version of claim 5, hence incorporates the corresponding rejections set forth in therein for the same reasons.

As per claim 18, refer to claim 6 for the same rationale.

6. Claims 3-4, 9-10, and 15-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stone, "In memory patching: three approaches", *(how to introduce breakpoints in an automated debugger and other marvels)*, March 1998, <http://fravia.anticrack.de/stone1.htm>, in view of Lillich, USPN: 5,619,698; further in view of Nowlin, Jr. et al., USPN: 6,484,309 (hereinafter Nowlin).

As per claim 3, Stone only disclose determining if a patching would be appropriate (pg. 2, 2nd para to pg. 3, top 2 para) by checking a CRC (pg. 2, bottom pg. 3 top) even though does not expressly specify

determining if at least one identifying attribute of a plurality of identifying attributes of the computer application program does not match at least one identifying attribute of a plurality of identifying attributes of compatible applications; and

if at least one of the identifying attributes matches, determining if the computer application program is incompatible, otherwise it is compatible.

But the concept of checking an identification attribute among other attributes of an suspicious application is disclosed. The reason why a code needs patching entails some identification and checking; and this is also evidenced further by Lillich who discloses a binding manager to see if the symbols for bind correctly into memory before investigating what patch is appropriate to fix the memory and enable the program to be linked for a given platform (Fig. 6-7; step 808 – Fig. 10; *fails the binding process, patches descriptor... variety of computing environments* - col. 10, lines 54 to col. 11, line 55). Analyzing attributes of application of a given platform and matching them so to determine if they are incompatible for such platform or OS executing environment is further shown via other methods of patching. Nowlin, also teaches patching of portions of memory using DLL, discloses a filter DLL to patch executable of one OS so to make appear compatible for another OS from analyzing memory space metrics or attributes and compatibility discrepancies (e.g. col. 4, lines 52 to col. 5, line 54); hence has disclosed matching attributes of one OS against what would be expected from another OS in order to establish compatible transition, i.e. matching compatible attributes against incompatible attributes. In the art of software distribution, the practice of matching properties of one executing environment with a the counterparts of a reference/target environment in order to establish compatible applicability of software leading to update or patching was a known concept

at the time the invention was made. Thus, it would have been obvious for one of ordinary skill in the art to apply the techniques so taught by Nowlin (e.g. matching metrics/attributes of an incompatible OS against those of a more compatible OS) to Lillich or Stone's determination for patching action; and would also be motivated to provide to Stone's method such matching suggested by Lillich or taught by Nowlin, by comparing such attributes against known incompatible application attributes because providing not only the known compliant (say one OS attributes) but also the known non-compliant patterns, i.e. incompatible attributes of another OS, would make the incompatibility checking even more adequate, for the benefits entailed from known concept to match attributes for compatibility checking known in many standard instances of software patching and version updating in software business.

As per claim 4, Nowlin further discloses storing identifying attributes of compatible applications (e.g. *entry point, table ...pointer in the file* – col. 6, lines 40 to col. 7, line 22 – Note: storing of known metrics and space address, e.g. start of code address, for a given OS reads on storing attributes of compatible application); and retrieving one of such stored identifying attributes for determining one of the attributes of the computer application program matches the stored attributes of the compatible applications (e.g. *startcode, version* – Fig. 10).

But Nowlin does not specify storing and using attributes of incompatible applications for the matching against attributes of the target application; but Nowlin's matching stored attributes of a compatible/target OS against incompatible OS is disclosing the same idea. This limitation would have been obvious by virtue of the rationale set forth in claim 3 above.

As per claim 9, this is the computer-readable medium version of claim 3, hence incorporates the corresponding rejections set forth in therein for the same reasons.

As per claim 10, this is the computer-readable medium version of claim 4, hence incorporates the corresponding rejections set forth in therein for the same reasons.

As per claim 15, in reference to claim 13, this is the computer system version of claim 3, hence incorporates the corresponding rejections set forth in therein for the same reasons.

As per claim 16, this is the system version of claim 4, hence incorporates the corresponding rejections set forth in therein for the same reasons.

Response to Arguments

7. Applicant's arguments with respect to claims 1-18 have been considered but are moot in view of the new ground(s) of rejection.

However, concerning remarks about Nowlin, whose method Applicants perceive as merely parsing an OS compatible application and translating the necessary program in that OS; it is also noted from Applicant further contend (Appl. Rmrks, pg. 10, middle paras) that Nowlin does not remotely teach or suggest a debugger. The current rejection approaches Nowlin as a dynamic process using Application DLL's to effect correct patches onto program memory to match a target OS environment as opposed to executing otherwise an application whose memory still bear the characteristics of an incompatible OS. The translation between memory parts or code binding time environment is analogous to any dynamic patching process; and that is the context under which Nowlin has been brought in to fulfill, such context being as in claims 3-4 for matching of compatible/incompatible OS parameters or attributes, as in the current rejection. The dynamic use of DLL for patching happens to be the main endeavor whereas starting a debugger is a second import because the rejection is purported to address another limitation, not the debugger limitation which the Stone reference has already met.

Conclusion

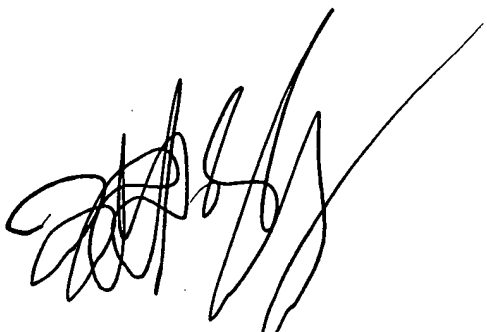
8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 703-872-9306 (for official correspondence) or redirected to customer service at 571-272-3609.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
November 24, 2004



TODD INGBERG
PRIMARY EXAMINER